AMENDMENTS TO THE CLAIMS

66. (Currently Amended) A parallel processing method of logic event simulation on circuits comprising a plurality of logic gates, the logic gates having interconnect lines therebetween, the method being carried out in a main processor and an associative memory mechanism, the associative memory mechanism comprising a plurality of associative arrays and at least one result register, ~~characterized in that there is provided~~ including an external memory, external to the associative memory mechanism, and means to transfer data between the associative memory and the external memory, the method comprising the steps of:

storing a circuit representation in external memory;

dividing the circuit representation into a plurality of circuit segments, at least one of the circuit segments containing a plurality of logic gates;

assigning a unique segment identifier to each segment;

generating a circuit segment table in the associative memory and storing the unique segment identifiers along with segment data in a circuit segment table;

for a time period, identifying segments that have a state S0 ~~are potentially active~~ in that time period based on the segment data stored in the circuit segment table;

bringing only the ~~potentially active~~ segments with a state S0, one at a time, into the associative memory mechanism from external memory for evaluation; and

evaluating the ~~active~~ segments with a state S0, in the associative memory mechanism and storing a result of the evaluation in a tangible medium.

67. (Previously Presented) The parallel processing method of logic event simulation as claimed in claim 66 in which the segment data stored in the circuit segment table comprises the maximum delay state of a segment, which indicates the maximum time delay in which any gate in the segment may make a transition.

68. (Previously Presented) The parallel processing method of logic event simulation as claimed in claim 66 in which the associative memory mechanism comprises a pair of associative arrays, associative array 1a and associative array 1b, an input value register bank and a hit list.

69. (Previously Presented) The parallel processing method of logic event simulation as claimed in claim 68 in which the circuit segment table data of associative array 1a, and associative array 1b and input value register bank are stored in external memory during segment evaluation.

70. (Previously Presented) The parallel processing method of logic event simulation as claimed in claim 66 in which after evaluation of a first segment, the segment data of all the first segments fan-out gates are updated.

71. (Previously Presented) The parallel processing method of logic event simulation as claimed in claim 70 in which when a new maximum delay state is greater than a previous

maximum delay state of a fan-out segment, the segment data is updated with the new maximum delay state.

72. (Previously Presented) The parallel processing method of logic event simulation as claimed in claim 66 in which inactive segments are not brought into the associative memory mechanism for evaluation until they have undergone an input change to a gate in that segment.

73. (Previously Presented) The parallel processing method of logic event simulation as claimed in claim 66 in which all interconnect lines are held in a segment dedicated to interconnect lines.

74. (Previously Presented) The parallel processing method of logic event simulation as claimed in claim 66 in which all logic gates of a particular type are held in segments with logic gates of the same type.

75. (Previously Presented) The parallel processing method of logic event simulation as claimed in claim 68 in which the segment table is an N_s x M bits segment table where N_s is equal to the number of segments and M is equal to the sum of the number of bits wide of associative array 1a, associative array 1b, input value register and the hit list.

76. (Previously Presented) The parallel processing method of logic event simulation as claimed in claim 68 in which at least portion of associative array 1a, associative array 1b, input value register bank and the hit list are used to store the segment table at all times.

77. (Previously Presented) The parallel processing method of logic event simulation as claimed in claim 68 in which when gate evaluations are completed for a particular time interval a previous segment table history is stored in the associative array 1b.

78. (Previously Presented) The parallel processing method of logic event simulation as claimed in claim 68 in which the input value register bank is shifted into associative array 1b, and associative array 1a contains a maximum delay state of each segment, test patterns are then applied to contents of array 1b to determine transitions to lower states.

79. [Cancelled]

80. (Previously Presented) The parallel processing method of logic event simulation as claimed in claim 66 in which the minimum delay state of all segments, $S_{STATEMIN}$, is calculated and all states are time advanced by $S_{STATEMIN}$ Time Units before evaluation of the segments commences.

81. (Previously Presented) The parallel processing method of logic event simulation as claimed in claim 68 in which the set up time, T_{SETUP}, of synchronous devices is modelled

where $T_{setup} = N.p + M$, where N = integer, M = integer < P and P = bit width of array 1b, where by the following steps are performed:-

the state entry of array 1a of a signal is set to S_{sn};

a start marker is placed in the left most position of array 1b;

array 1b is incremented in time and when start marker reaches the right-most position of array 1b and the signal has remained constant the state S_{sn} is decremented to S_{sn-1}, and the next time array 1b is incremented the start marker is returned to the left-most position in array 1b once again and array 1b is then incremented in the normal manner;

the previous step is repeated until state entry $S_{sn} = S_{so}$, then the array 1b is incremented another M times; and

if the signal has remained constant for N.p + M time units then the state entry in array 1a is set to state setup, S_{SETUP}.

82. (Currently Amended) The parallel processing method of logic event simulation as claimed in claim 68 in which the hold time, T_{HOLD}, of a synchronous device is modelled where $T_{hold} = R.p + Q$, and where R = integer, Q = integer < P and P = bit width of array 1b using the following steps:

when a clock makes a transition there is a search of state entries in array 1a to see ~~if any~~ if an input signal is in state S_{setup};

any input signal in state S_{setup} is updated to the state S_{HR}, and a start marker is placed in the left-most position of array 1b, array 1b is incremented in the normal manner and until the

marker has made its way to the right-most position in array 1b and the signal has remained constant;

the state is decremented to S_{HR-1} , the start marker is returned to the left-most position in array 1b the next time that array 1b is incremented, array 1b is then incremented and this is continued until the state is equal to S_{Ho};

when $S_{HR} = S_{HO}$ then the array 1b is incremented a further Q times;

on the signal remaining constant over the entire period then the signal state is updated to S_{HOLD} and an output value of the synchronous device is ascertained.

83. (Previously Presented) The parallel processing method of logic event simulation as claimed in claim 82 in which if the output value of the device has changed it is propagated to the fan-out list of the device.

84. (Previously Presented) The parallel processing method of logic event simulation as claimed in claim 81 in which successive states are generated by causing a shift right operation in array 1a.

85. (Previously Presented) The parallel processing method of logic event simulation as claimed in claim 66 in which there is provided an amended result registering mechanism in which when a number of tests are carried out on a gate pair a result of each test is sent to a result register where on completion of all the tests the result register will indicate that all tests were successful or that at least one was unsuccessful.

86. (Previously Presented) The parallel processing method of logic event simulation as claimed in claim 85 in which the result register comprises an adder.

87. (Previously Presented) The parallel processing method of logic event simulation as claimed in claim 85 in which the result register is a bi-state device.

88. (Previously Presented) The parallel processing method of logic event simulation as claimed in claim 85 in which the result register comprises a D-flip flop.

89. (Previously Presented) The parallel processing method of logic event simulation as claimed in claim 87 in which the result register on start-up is supplied with a priming input instead of last result so that the result register is ready to receive a first result.

90. (Previously Presented) The parallel processing method of logic event simulation as claimed in claim 85 in which the amended result registering mechanism further comprises a result polarity circuit to invert a result and ensure a logic 1 is applied to the result register when the correct response to a test was for the test to be failed.

91. (Previously Presented) The parallel processing method of logic event simulation as claimed in claim 90 in which the result polarity circuit comprises a pair of AND gates, a pair of inverters and an OR gate, a result polarity control is fed to each of the AND gates, the other

input of each of the AND gates being provided by the result of a test carried out on a gate pair, the inverters inverting the two inputs to one of the AND gates, the outputs of the AND gates being fed directly to the OR gate.

92. (Previously Presented) The parallel processing method of logic event simulation as claimed in claim 85 in which the amended result registering mechanism is further provided with a logic combination circuit to determine whether an output gate pair of array 1b are ANDed or ORed together.

93. (Previously Presented) The parallel processing method of logic event simulation as claimed in claim 92 in which the logic combination circuit further comprises three AND gates and a logic combination circuit control, the logic combination circuit control being anded individually with each output of an array 1b gate pair and the gate pair anded in the third AND gate, when the logic combination requires an AND operation to be carried out, logic combination circuit control is given a value 0 and if an OR operation is required logic combination circuit control is given a logic value 1.

94. (Currently Amended) The parallel processing method of logic event simulation as claimed in claim 93 in which the logic combination circuit further comprises an OR gate, each of the outputs of the three AND gates being fed to the OR gate.

95. (Previously Presented) The parallel processing method of logic event simulation as claimed in claim 94 in which the output of the OR gate is led to the result polarity circuit as the result of a test carried out on a gate pair.

96. (Previously Presented) The parallel processing method of logic event simulation as claimed in claim 85 in which there is provided an amended result registering mechanism for each gate pair.

97. (Currently Amended) The parallel processing method of logic event simulation as claimed in claim 66 in which when all ~~active~~ segments have a state > S0, a check of all segment states is made until the lowest segment state S_{min} is found, then each segment state is decremented by S_{min} in order to advance simulation to the next evaluation stage.

98. (Previously Presented) The parallel processing method of logic event simulation as claimed in claim 97 in which the lowest state value is stored in a low global register and each time there is a gate state change, if a new state is less than the lowest state value, the lowest state value in the low global register is replaced by the new state.

99. (Previously Presented) The parallel processing method of logic event simulation as claimed in claim 66 in which there is provided a scan system comprising a priority decoder and a shift register.

100. (Previously Presented) The parallel processing method of logic event simulation as claimed in claim 66 in which there is provided a segment address table and the segment address table is divided into a number of rows, each row being M bits long, and each segment address is stored in the M – D bits of the segment row when the number of segments = 2^D.

101. (Currently Amended) A processor for parallel processing of logic event simulation on circuits comprising a plurality of logic gates, the logic gates having interconnect lines therebetween, the processor further comprising a main processor and an associative memory mechanism, the associative memory mechanism comprising a plurality of associative arrays and at least one result register, ~~characterized in that~~comprising:

~~there is provided~~ an accessible external memory, external to the associative memory mechanism, for storage of a circuit representation;

means to divide the circuit representation into a plurality of circuit segments at least one of the circuit segments containing a plurality of logic gates;

means to allocate a circuit segment identifier to each circuit segment, each circuit segment identifier having circuit segment data associated therewith;

the associative memory mechanism comprising a segment table for storage of segment identifiers and segment data;

means to identify ~~active~~ segments that have a state S0 in any one time interval based on the segment data; and

means to retrieve only those ~~active~~ segments with a state S0, one at a time, from external memory for evaluation by the associative memory mechanism and means to evaluate the ~~active~~

segments with a state S0 in the associative memory mechanism and means to store the result of the evaluation on the ~~active~~ segments in a tangible medium.

102. (Previously Presented) The processor as claimed in claim 101 in which the segment data further comprises the maximum time delay for a gate in that segment to undergo a transition.

103. (Previously Presented) The processor as claimed in claim 101 in which the associative memory mechanism further comprises a pair of associative arrays (1a and 1b), an input value register and a hit list.

104. (Previously Presented) The processor as claimed in claim 103 in which cache data of associative array 1a and associative array 1b are stored in external memory during evaluation.

105. (Previously Presented) The processor as claimed in claim 101 in which after evaluation of the segment data the segments fan-out lists are updated.

106. (Previously Presented) The processor as claimed in claim 101 in which all interconnect lines are held in a segment dedicated to interconnect lines.

107. (Previously Presented) The processor as claimed in claim 101 in which all logic gates of a particular type are held in segments with logic gates of the same type.

108. (Previously Presented) The processor as claimed in claim 103 in which the segment table is an N_S x M bit segment table where N_s is equal to the number of segments and M is equal to the sum of all the number of bits width of associative array 1a, associative array 1b, input value register and the hit list.

109. (Previously Presented) The processor as claimed in claim 103 in which the set up time, T_{SETUP}, of synchronous devices is modelled and in which $T_{SETUP} = N.p + M$, where N is equal to an integer, M is equal to an integer < P, and P is equal to the bit width of array 1b, in which:

the state entry in array 1a of this signal is set to S_{SN};

a start marker is placed in the left-most position of array 1b;

array 1b is incremented in the normal manner and when the start marker reaches the right-most position of array 1b and the signal has remained constant the state S_{SN} is decremented to S_{SN-1}, and the next time array 1b is incremented the start marker is returned to the left-most position in array 1b once again and the array 1b is then incremented in the normal manner;

the previous step is repeated until state entry = S_{S0}, then the array 1b is incremented another M times; and

when the signal has remained constant for N.p + M time units then the state entry in array 1a is set to state setup, S_{SETUP}.

110. (Previously Presented) The processor as claimed in claim 109 in which the hold time, T_{HOLD} , of synchronous devices is modelled where $T_{HOLD} = R.p + Q$, and where R is an integer, Q is an integer < P and P is equal to the bit width of array 1b, in which: -

when a clock makes a transition there is a search of state entries in array 1a to see if any input signals are in state S_{SETUP} ;

states in S_{SETUP} are updated to the state S_{HR} , and a start marker is placed in the left-most position of array 1b, array 1b is incremented in the normal manner until the start marker has made its way to the right-most position in array 1b and the signal has remained constant;

the state is decremented to S_{HR-1} , and the next time array 1b is incremented start marker is returned to the left-most position in array 1b, array 1b is then incremented in the normal manner and this is continued until the state is equal to S_{H0} ;

when S_{HR} equals S_{H0} then the array 1b is incremented a further Q times;

when signals remain constant over the entire period then the signal state is updated to S_{HOLD} and the output value of the device is ascertained.

111. (Previously Presented) The processor as claimed in claim 110 in which when the output value has changed, it is propagated to the fan-out list of the device.

112. (Previously Presented) The processor as claimed in claim 101 in which there is provided an amended result registering mechanism in which when a number of tests are carried out on the gate pair the result of each test is sent to a result register where on completion of all

the test the result register output indicates that all tests are successful or that at least one test was unsuccessful.

113. (Previously Presented) The processor as claimed in claim 112 in which the result register comprises an adder.

114. (Previously Presented) The processor as claimed in claim 112 in which the result register is a bi-state device.

115. (Previously Presented) The processor as claimed in claim 112 in which the result register comprises a D-flip flop.

116. (Previously Presented) The processor as claimed in claim 114 in which the result register on start-up is supplied with an appropriate priming input instead of last result so that it is ready to receive a first actual result.

117. (Previously Presented) The processor as claimed in claim 112 in which the amended result registering mechanism further comprises a result polarity circuit to invert a result and ensure logic 1 is applied to the result register when the correct response to a test was that a particular test on a gate pair was failed.

118. (Previously Presented) The processor as claimed in claim 112 in which the amended result register mechanism is further provided with a logic combination circuit to determine whether an output gate pair of array 1b are ANDed or ORed together.

119. (Previously Presented) The processor as claimed in claim 118 in which logic combination circuit further comprises three AND gates and a logic combination circuit control, the logic circuit control being anded individually with each output of array 1b gate pair, and the gate pair ANDed in the third AND gate, when the logic combination requires an AND operation, logic combination circuit control is given a value 0 and when an OR operation is required logic combination circuit control is given a logic 1.

120. (Previously Presented) The processor as claimed in claim 119 in which the logic combination circuit further comprises an OR gate, each of the outputs of the three and gates being fed to the OR gate.

121. (Previously Presented) The processor as claimed in claim 120 in which the output of the OR gate is led to the result polarity circuit.

122. (Currently Amended) The processor as claimed in claim 101 in which when all ~~active~~ segments have a state > S0, a check of all segment states is made until the lowest segment state S_{min} is found, then each segment state is decremented by S_{min} in order to advance simulation to the next evaluation stage.

123. (Previously Presented) The processor as claimed in claim 122 in which the lowest state value is stored on a low global register and each time there is a gate state change, when the new state is less than the low global state register value, the new state replaces the low global state register value.

124. (Previously Presented) The processor as claimed in claim 101 in which there is provided a scan system comprising a priority decoder and a shift register.

125. (Previously Presented) The processor as claimed in claim 112 in which there is provided an amended result registering mechanism for each gate pair.

126. (Previously Presented) The processor as claimed in claim 101 in which there is provided a segment address table and the segment address table is divided into a number of rows, each row being M bits long, and each segment address is stored in the M-D bits of the segment row when the number of segments is equal to 2^D.

127. (Currently Amended) The processor as claimed in claim 101 in which the processor is embodied in computer readable format and stored in a physical carrier.

128. (Previously Presented) The processor as claimed in claim 127 in which the computer readable format is stored on a disc.

129. (Previously Presented) The processor as claimed in claim 127 in which the computer readable format is stored on a record medium.

130. (Cancelled)